Problem 1:

(a) True
(b) False
(c) False
(d) False
(e) True
(f) False
(g) False
(h) True

Problem 2:

(a) In order to prove that the 4-SAT problem is NP-Hard, deduce a reduction from a known NP-Hard problem to this problem. Deduce a reduction from which the 3-SAT problem can be reduced to the 4-SAT problem. For each clause of the 3-SAT formula f, for example, a literal a and its corresponding complement a' should be added to the formula. Let there be a clause c, such that c = u V v' V w . To convert it in 4-SAT, we convert c to c', such that,
c' = (u V v' V w V a) AND (u V v' V w V a').
After simulating this conversion, two properties hold :

- If 3-SAT has a satisfiable assignment, which means, every clause evaluates to true for a specific set of literal values, then 4-SAT will also hold, because each clause-set is formed by a combination of a literal and its complement, whose value won't make any difference.
- If 4-SAT is satisfiable for any (u V v V w V a) and (u V v V w V a'), then 3-SAT is also satisfiable because a and a' are complement, which indicates that the formula is satisfiable due to some other literal except a too.

Therefore, following the above propositions, the 4-SAT problem is NP-Complete.

(b) The algorithm is as follows:

```
Init a an undirect graph G with (V,E), and initial all vertices are all false.
For item_i in vertices in CNF
    ● If item_i is true then change vertical_i in G to true
If All vertices in G are all true then return true
Else return false.
```

Problem 3:

Algorithm Describe:

Incrementally add to G in such a way that at each step we learn the coloring of one vertex. Here we add a triangle of vertices (so they must all be different colors) and then, for each vertex, try connecting it to every combination of 2 of those new vertices (so its color will be the same as the new vertex it is not connected to). The algorithm is as follows:

```
If G is not 3-colorable then return null
G <- G with new vertices r, g, b and edges (r, g), (g,b), (b,r)
For each vertex v_i
```

- G_r <- G’ with edges (v_i, g), (v_i, b)
- G_g <- G’ with edges (v_i, r), (v_i, b)
- G_b <- G’ with edges (v_i, r), (v_i, g)
- G’ <- whichever of G_r, G_g, G_b is 3-colorable

For each vertex v, the color of v is the one of r, g, b it is not connected to in the final G’.